PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
(fraunh01.032)

Applicant: Luo, et al.	**Confirmation No.:** 8663
Application No: 10/019828	**Group Art Unit:** 2136
Filed: 5/21/2002	**Examiner:** Baum, Ronald

Title: *Obfuscation of executable code*

Assistant Commissioner for Patents
Washington, DC 20231

Response to a non-final Office action under 37 C.F.R. 1.111

Summary of the prosecution

This application is the U.S. National Stage of PCT/US00/13128. A *Demand for an International Preliminary Examination* under PCT Chapter II was made in this application and claims 1-13 submitted for examination were found to meet the requirements of PCT Article 33(2-4). The Application entered the National Stage on 5/21/02 and Examiner mailed a first Office action on 6/21/06. In that Office action, Examiner objected to claims 6-8 under 37 C.F.R. 1.75(c) because of failure to use the proper form in specifying multiple dependencies and rejected claims 1-13 under 35 U.S.C. 102(b) as anticipated by USSN 5,778,071, Caputo, et al., *Pocket encrypting and authenticating communications device*, issued 7/7/98, henceforth "Caputo". Applicants amended their claims 6-8 to overcome the objection to claims 6-8 and traversed the rejection under 35 U.S.C. 102(b). As part of the traversal, Applicants called Examiner's attention to three references which were cited but not applied by Examiner but are in fact more relevant to Applicants' claims than Caputo. Among these references was U.S. Patent 6,668,325, Collberg, et al., *Obfuscation techniques for enhancing software security*, issued Dec. 23, 2003 and claiming priority from a New Zealand patent application filed June 9, 1997 (henceforth "Collberg"). Applicants cited the published PCT application WO99/0815, which is based on the New Zealand application, in their *Description of related art.*

Examiner mailed a second non-final Office action in the above application on Dec. 19, 2006 in which he persisted in his rejection of claims 6-8 as amended under 37 CFR 1.75(c) and rejected claims 1-13 under 35 U.S.C. 102(e) as anticipated by Collberg. Applicants are traversing both the objections under 37 C.F.R. 1.75(c) and the rejections under 35 U.S.C. 103(e).

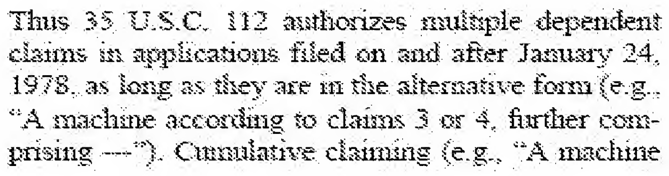
Traversal

The objections to claims 6-8

These claims are multiply-dependent. As amended in Applicants' response of October 20, 2006, the language in the preamble that sets forth the multiple dependency reads as follows:

> The method of executing obfuscated code set forth *in any one of claims 3, 4, or 5* wherein: (emphasis added)

MPEP 6801(n) sets forth the requirements for multiply-dependent claims. It currently reads as follows:



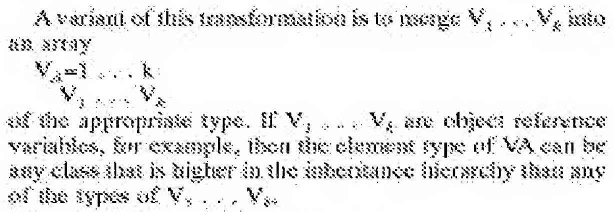
> Thus 35 U.S.C. 112 authorizes multiple dependent claims in applications filed on and after January 24, 1978, as long as they are in the alternative form (e.g., "A machine according to claims 3 or 4, further comprising ---"). Cumulative claiming (e.g., "A machine

Applicants' attorney respectfully submits that the language "in any one of claims 3, 4, or 5" does set forth the multiple dependencies "in the alternative form", as required by MPEP 6801(n). Applicants' attorney believes that the language sets forth the multiple dependencies in the alternative form as a matter of English grammar and is confirmed in this belief by the fact that he has been writing multiply-dependent claims in this form for 27 years now without objection from the USPTO.

Traversal of the rejection of claim 1

Claim 1 is directed to the method of data field obfuscation described at page 6, line 31 through page 7, line 14 and shown at 301 in FIG. 3. As set forth in the claim, the method "obfuscat[es] executable code that uses a first reference including a symbolic object name and a symbolic field name to reference a field containing data". The "first reference" is embodied in the `person` symbolic object name of FIG. 2 and the `string` and `person` symbolic field names used in FIG. 2 to reference fields containing the `person` object's data. In FIG. 3, `person` is defined using the vector v, whose data fields contain the `person` object's data. In v, the data fields are referenced by array indices, and v is thus an embodiment of the "second reference that references the field by the defined object's name and the field as required by the defined object". A person object is now created by creating an instance of the array v that has two elements, one for each of the two variables (305), and then using the index of the variable's element in the array to reference the variable's elements, as shown at 307 and 309.

The closest Collberg comes to any of this is col. 24, lines 53-60, which reads as follows:



A variant of this transformation is to merge $V_1 \ldots V_k$ into an array

$V_{A}=1 \ldots k$

$V_1 \ldots V_k$

of the appropriate type. If $V_1 \ldots V_k$ are object reference variables, for example, then the element type of VA can be any class that is higher in the inheritance hierarchy than any of the types of $V_1 \ldots V_k$.

In this, transformation, the variables simply become elements of the array; there is no notion of using the technique to obfuscate the relationship between an object and its fields, and consequently nothing like the claimed steps of

> defining an object wherein the field is not referenced by a symbolic field name; and
>
> replacing the first reference with a second reference that references the field by the defined object's name and the field as required by the defined object.

Because that is the case, Collberg does not show all of the limitations of claim 1 and the rejection is without foundation.

Traversal of the rejection of claim 2

Claim 2 is directed to an obfuscation method that may be used with code written in languages such as Java that have *reflection*. As set forth beginning at page 8, line 16 languages that have reflection have mechanisms for returning information about the language's classes at run time. In Java, the mechanism is methods in the Java system classes that return class information about Java objects. A method for using reflection for obfuscation is described beginning at page 8, line 31 and illustrated at FIG. 5.

Collberg says nothing at all about Java's reflection mechanism. This is apparent from the results of a Lexis search on `patno=(6,668,325) and (reflect!)`, set up to display 25 words on each side of the search term. What it returned was the following:

...low execution time penalty should be favored. This latter point is accomplished by selecting transformations that maximize potency and resilience, and minimize cost.

An obfuscation priority is allocated to a source code object. This will **reflect** how important it is to obfuscate the contents of the source code object. For example, if a particular source code object contains highly sensitive proprietary material, then the obfuscation priority will be high. An execution ...

...AcceptCost.

c) An updated obfuscation priority mapping I.

d) A Boolean return value which is TRUE if the termination condition has been reached.

The Done function serves two purposes. It updates the priority queue I to **reflect** the fact that the source code object S has been obfuscated, and should receive a reduced priority value. The reduction is based on a combination of the resilience and potency of the transformation. Done also updates Reqobf and AcceptCost, and determines whether the termination condition has been ...

As will be immediately apparent from the search results, the term "reflect" is not used in Collberg to refer to the Java reflection mechanism. Since that is the case, Collberg does not disclose the limitations of claim 2 and therefore does not anticipate it.

Traversal of the rejection of claims 3-12

These claims all concern methods of using encryption to obfuscate constructs whose definitions are external to the executable code. In Java, the constructs that are defined in Java system classes are constructs which are external to the executable code. In claim 11, such constructs are set forth as "constructs belonging to an execution environment in which the executable code will execute". The claimed methods are supported by the description which begins at page 9, line 26 and by FIG. 6.

Collberg discloses no more about using encryption to obfuscate Java system constructs than he does about using reflection to obfuscate. A Lexis search on `patno=(6,668,325) and (encrypt!)`, set up to display 25 words on each side of the search term, returned the following:

SUMMARY:

...structures, abstractions, and organization of the code.

Software patents provide more comprehensive protection. However, it is clearly an advantage to couple legal protection of software with technical protection.

Previous approaches to the protection of proprietary software have either used **encryption-**based hardware solutions or have been based on simple rearrangements of the source code structure. Hardware-based techniques are non-ideal in that they are generally expensive and are tied to a specific platform or hardware add-on. Software solutions ...

DRWDESC:

...providing software security by (a) server-side execution and (b) partial server-side execution;

FIGS. 4a and 4b show techniques for providing software security by (a) using **encryption** and (b) using signed native code;

FIG. 5 shows a technique for providing software security through obfuscation;

FIG. 6 illustrates the architecture of an example of an obfuscating tool suitable for ...

DETDESC:

...locally on the user's site, and a private part (that contains the algorithms that Alice wants to protect) that is run remotely, for example, as shown in FIG. 3b.

Another approach would be for Alice to **encrypt** her code before it is sent off to the users, for example, as shown in FIG. 4a. Unfortunately, this only works if the entire decryption/execution process takes place in hardware. Such systems are described ...

...available deobfuscation algorithms, and (c) the amount of resources (time and space) available to the deobfuscator. Ideally, we would like to mimic the situation in current public-key cryptosystems, in which there is a dramatic difference in the cost of **encryption** (finding large primes is easy) and decryption (factoring large numbers is difficult). We will see that there are, in fact, obfuscating transformations that can be applied in polynomial time but which require exponential time to ...

...FIG. 31. For an overview of the opaque constructs that have been discussed above, see FIG. 32. However, the present invention should not be limited to the exemplary transformations and opaque constructs discussed above.

11.1 The Power of Obfuscation

> **Encryption** and program obfuscation bear a striking resemblance to each other. Not only do both try to hide information from prying eyes, they also purport to do so for a limited time only. An encrypted document has a limited shelf-life: it is safe only for as long as the **encryption** algorithm itself withstands attack, and for as long as advances in hardware speed do not allow messages for the chosen key-length to be routinely decrypted. The same is true for an ...

As is clear from the description of FIG. 4a at col. 9, lines 55-67, what is described there is the technique of simply encrypting all of the downloaded code. The problem with this technique is of course that if the code is decrypted and then executed, it can be reverse engineered by simply decompiling it. The other references are simply to decryption generally. Because there is no disclosure whatever in Collberg of the use of encryption for obfuscation, the reference cannot anticipate claims 1, 9, and 11. Further, the additional limitations of dependent claims 4-8 and 10 all involve encryption, and are consequently patentable in their own rights over Collberg. Claim 13 is a dependent Beauregard claim, and as such, its patentability depends completely on the patentability of the claims it is dependent from. As just pointed out, each of claims 1, 2, 3, 9, and 11 is patentable, and consequently so is claim 13.

Conclusion

Applicants have traversed all of Examiner's objections and rejections and have thereby been completely responsive to Examiner's Office action of Dec. 19, 2006 as required by 37 C.F.R. 1.111(b) and respectfully request that Examiner continue with his examination as provided by 37 C.F.R. 1.111(a). No fees are believed to be required for this response. Should any be, please charge them to Deposit Account #501315.

Respectfully submitted,

/Gordon E. Nelson/
Attorney of record,
Gordon E. Nelson
57 Central St., P.O. Box 782
Rowley, MA, 01969,
Registration number 30,093
Voice: (978) 948-7632
Fax: (617) 788-0932
3/19/07
Date